EMBEDDED
LINUX
CONFERENCE

@

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Day-to-day testing of Linux *'next'* kernel branch - my story

Marek Szyprowski

#ossummit

THE LINUX FOUNDATION

Introduction – who am I?

- Marek Szyprowski

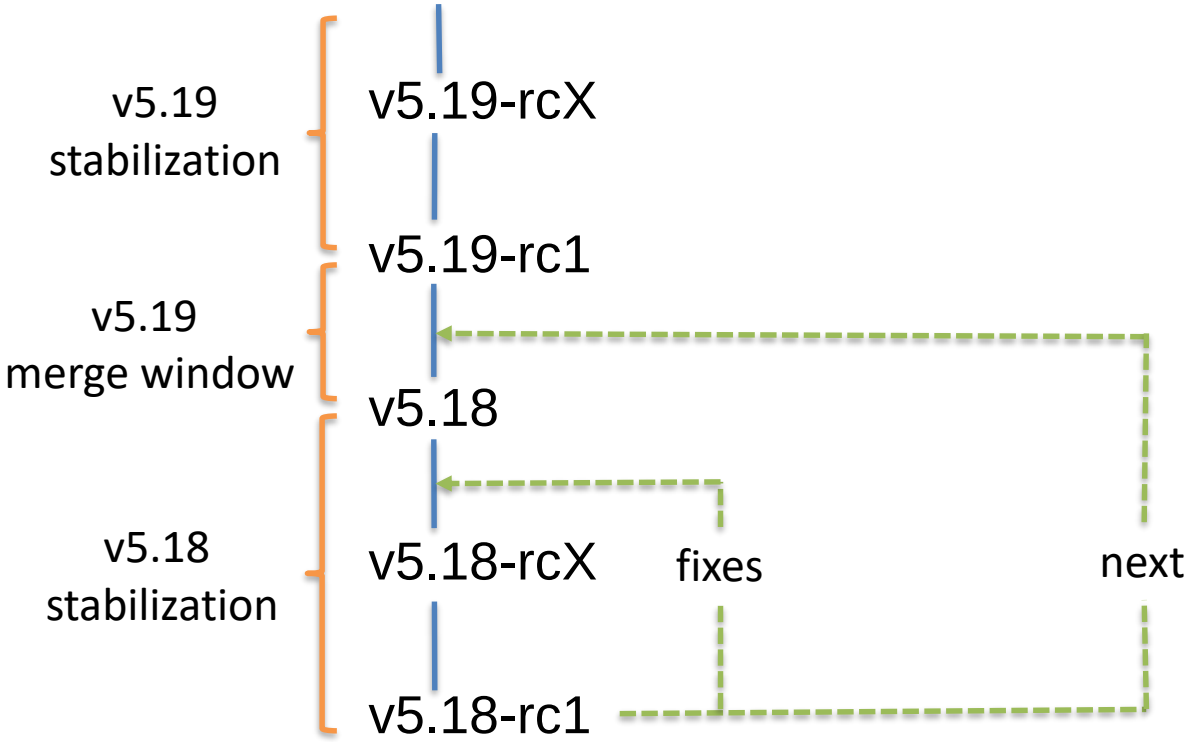
- m.szyprowski@samsung.com
- Samsung R&D Institute Warsaw, Poland since 2008

- Linux kernel developer since 2009
- Linux kernel maintainer since 2011
- Day-to-day testing of Linux kernel since 2018

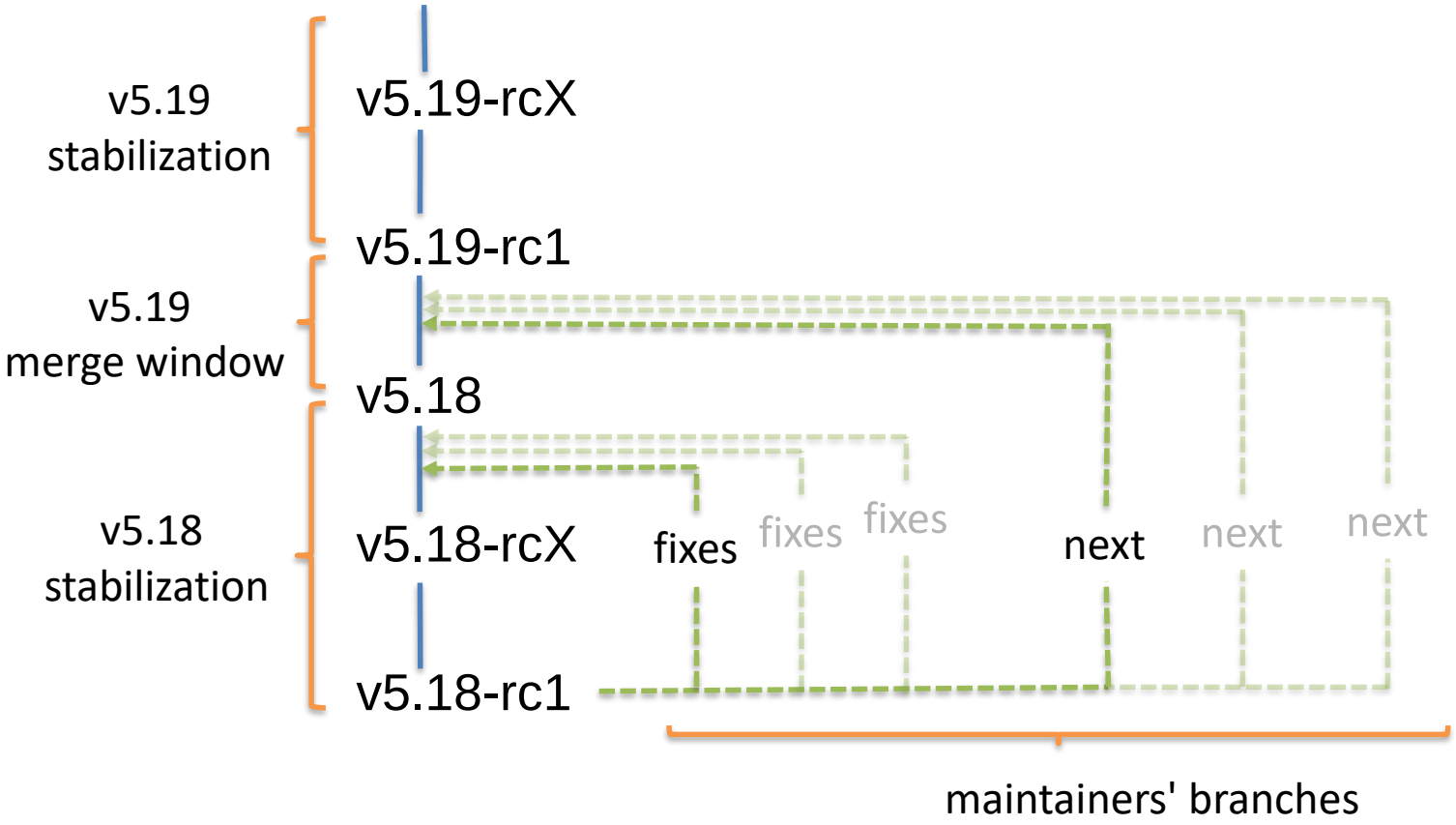
Linux kernel – development model

- Release every 3 months
- Hierarchical maintenance model
- Maintainers manage their ‘fixes’ and ‘next’ branches
- New code
  - Must be first tested in the ‘next’ branch
  - Merged during ‘2 weeks merge window’
  - Typically stabilized during -rc period
- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/

Linux kernel – development model



Linux kernel – development model



Linux-next

- Contains all ‘next’ branches merged together
- Released almost every work day
- Goal: check for regressions before they reach main kernel branch
- https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git/

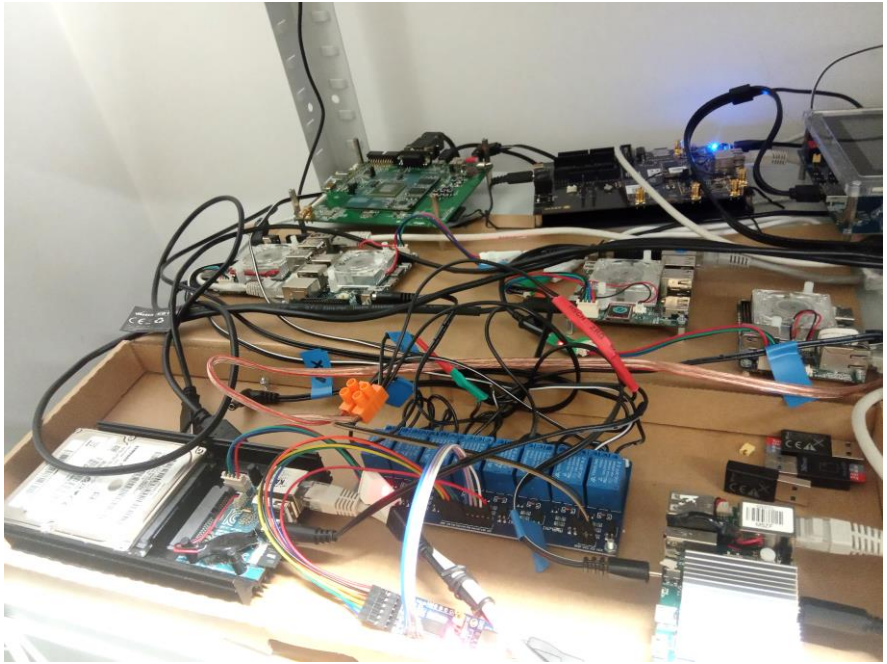
Kernel ‘testing’

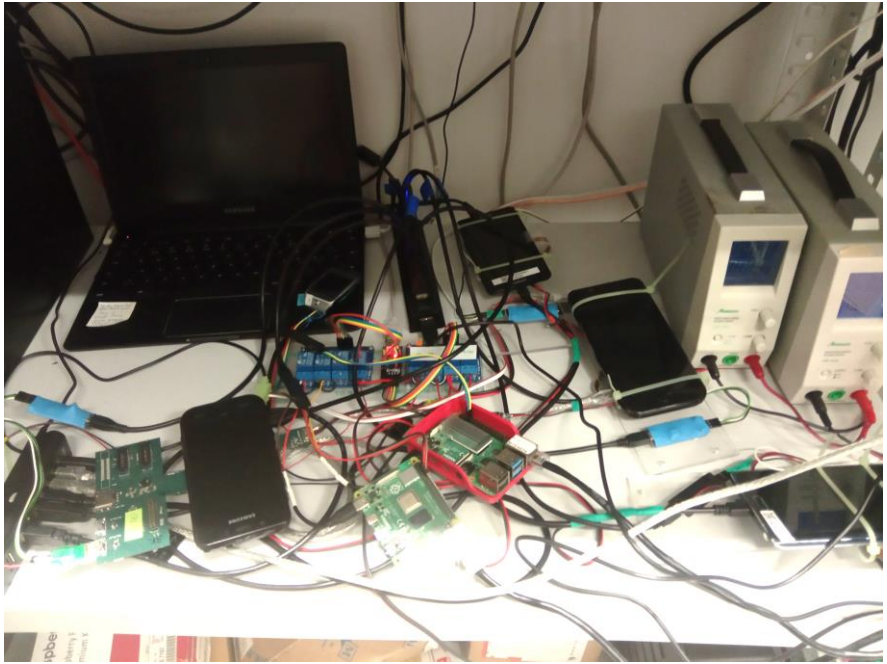
- Many levels of ‘testing’ possible
  - Compile-time
  - Booting on QEmu
  - Booting on real hardware
  - Running specific user-space tools
  - Advanced test scenarios

My test farm – overview

- 30 SBCs connected to the standard PC
  - All ARM 32bit and 64bit SoC based
  - Exynos based boards: Odroid family (like U3, XU4, …), Tizen reference boards (Trats, Trats2, TM2e), Chromebooks (Snow, Peach)
  - Others: Raspberry Pi 3B+, 4B+, Amlogic Meson based Odroid family (C4, N2), Khadas VIM3/VIM3l, Qualcomm Dragon Board 410c, ARM Juno R1
- Over 50 USB devices connected, 2 Ethernet switches, 10 USB hubs, …
- Occupies 4 storage shelfs in the test room, lots of cables…

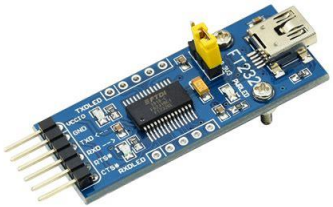
My test farm – a few pictures





My test farm – hardware

- UART for the kernel logs & user console
- Ethernet for the connectivity
  - Built-in
  - USB dongle
  - CDC USB gadget
- Power controlled with USB FT232RL adapters and a set of relays
- A few USB cameras for monitoring boards display



My test farm – software

- Kernel & modules loaded via TFTP from the PC
- Debian rootfs stored on the board's built-in medium (eMMC, SD)
- UART adapters identified by USB '*serial-id*' feature
- Accessible via SSH on the PC
  - single script to control power or reset board
  - get access to console device for the specified board

- No board reservation or sharing management
- Main goal is to allow quick access to all boards

Boot test

- First test ‘on the real hardware’

- Can be easily done with a shell script:
  - Compile kernel
  - Deploy compiled kernel & modules
  - Reset/power on board
  - Wait for the login prompt (‘*expect’* tool)

- Even such simple tests allowed me to report a few issues.

More tests

- Usually one tries to test a bit more in any manual test
- Run tools like *modetest*, *ifconfig*, *ping*, *rtcwake*, …
- Inspect */sys/kernel/debug/devices_deferred*
- Check the results on board under the test
- Use various kernel configuration files: arm/exynos_defconfig, arm/multi_v7_defconfig, arm64/defconfig
- A single shell script for everything becomes a problem

My testing solution

- Made script a bit more generic
- Data loaded from separate files
  - Configs: list of boards, arch, kernel defconfig, cross-copiler
  - Test: ‘*expect’* rules (send characters, wait for a given phrase)

My testing solution

- Made script a bit more generic
- Data loaded from separate files
  - Configs: list of boards, arch, kernel defconfig, cross-copiler
  - Test: '*expect*' rules (send characters, wait for a given phrase)
- Nice coloured output

File Edit View Search Terminal Help

```
arm_exynos:
       Test:   boot   bash   defer   rtc0   rtc1   drm    fbtest  ping   wifi   bt     alsa   cpu_hp  sleep  sleep2  ping2  reboot  off    status  [summary]
Board artik5:  okay   okay   okay    okay   okay   n.a.   n.a.    okay   okay   okay   n.a.   okay    n.a.   n.a.    okay   okay    okay   done    [okay]
Board rinato:  okay   okay   okay    okay   okay   okay   okay    okay   n.a.   okay   n.a.   okay    okay   okay    okay   okay    okay   done    [okay]
Board c210:    okay   okay   okay    okay   okay   okay   okay    fail   n.a.   okay   n.a.   n.a.    n.a.   n.a.    fail   n.a.    okay   done    [okay]
Board trats:   okay   okay   okay    okay   okay   okay   okay    okay   fail   okay   n.a.   okay    okay   okay    okay   okay    okay   done    [okay]
Board u3:      okay   okay   okay    okay   okay   okay   okay    okay   n.a.   n.a.   okay   okay    okay   okay    okay   okay    okay   done    [okay]
Board u3sd:    okay   okay   okay    okay   okay   okay   okay    okay   n.a.   n.a.   okay   okay    okay   okay    okay   okay    okay   done    [okay]
Board trats2:  okay   okay   okay    okay   okay   okay   okay    fail   okay   okay   okay   okay    okay   okay    okay   okay    okay   done    [WARN]
Board snow:    okay   okay   okay    okay   okay   okay   okay    okay   okay   okay   okay   okay    okay   fail    okay   n.a.    okay   done    [okay]
Board arndale: okay   okay   okay    okay   okay   okay   okay    okay   n.a.   n.a.   okay   okay    okay   fail    fail   fail    fail   done    [okay]
Board aocta:   okay   okay   okay    okay   okay   okay   n.a.    okay   n.a.   n.a.   n.a.   n.a.    n.a.   n.a.    okay   okay    okay   done    [okay]
Board pit:     okay   okay   okay    okay   okay   okay   okay    okay   okay   okay   okay   okay    okay   okay    okay   n.a.    okay   done    [okay]
Board xu3:     okay   okay   okay    okay   okay   okay   fail    okay   n.a.   n.a.   okay   okay    okay   okay    okay   okay    okay   done    [okay]
Board xu3lite: okay   okay   okay    okay   okay   okay   okay    okay   n.a.   n.a.   okay   okay    okay   okay    okay   okay    okay   done    [okay]
Board xu4:     okay   okay   okay    okay   okay   okay   okay    okay   n.a.   n.a.   okay   okay    okay   okay    okay   okay    okay   done    [okay]
Board hc1:     okay   okay   okay    okay   okay   n.a.   n.a.    okay   n.a.   n.a.   n.a.   okay    okay   okay    okay   okay    okay   done    [okay]
Board pi:      okay   okay   okay    okay   okay   okay   okay    okay   okay   okay   okay   okay    okay   okay    okay   n.a.    okay   done    [okay]
```

Simple regression

- Linux next-20220308
- Lets run the tests…

Simple regression

- Linux next-20220308
- Lets run the tests

```
File  Edit  View  Search  Terminal  Help
202207201323-next-20220308
Building arm (exynos_defconfig + no ARM_EXYNOS_CPUIDLE)):       okay

arm_exynos_nocpuidle:
       Test:   boot    bash    defer   rtc0    rtc1    drm     fbtest  ping    wifi    bt      alsa    cpu_hp  sleep   sleep2  ping2   reboot  off     status  [summary]
Board artik5:  okay    okay    okay    okay    okay    n.a.    n.a.    okay    okay    okay    n.a.    okay    n.a.    n.a.    okay    okay    okay    done    [WARN]
Board rinato:  fail    fail    fail    fail    fail    fail    fail    fail    n.a.    fail    n.a.    fail    fail    fail    fail    fail    fail    done    [WARN]
Board c210:    fail    fail    fail    fail    fail    fail    fail    fail    n.a.    fail    n.a.    n.a.    n.a.    n.a.    fail    n.a.    fail    done    [WARN]
Board trats:   fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    n.a.    fail    fail    fail    fail    fail    fail    done    [WARN]
Board u3:      fail    fail    fail    fail    fail    fail    fail    fail    n.a.    n.a.    fail    fail    fail    fail    fail    fail    fail    done    [WARN]
Board trats2:  fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    done    [WARN]
Board arndale: fail    fail    fail    fail    fail    fail    fail    fail    n.a.    n.a.    fail    fail    fail    fail    fail    fail    fail    done    [WARN]
Board aocta:   okay    okay    okay    okay    okay    okay    n.a.    okay    n.a.    n.a.    n.a.    n.a.    n.a.    n.a.    okay    okay    okay    done    [WARN]
Board pit:     fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    n.a.    fail    done    [WARN]
Board xu3:     okay    okay    okay    okay    okay    okay    fail    okay    n.a.    n.a.    okay    okay    okay    okay    okay    okay    okay    done    [WARN]
Board xu3lite: okay    okay    okay    okay    fail    fail    fail    fail    n.a.    n.a.    fail    fail    fail    fail    fail    fail    fail    done    [WARN]
Board xu4:     fail    fail    fail    fail    fail    fail    fail    fail    n.a.    n.a.    fail    fail    fail    fail    fail    fail    fail    done    [WARN]
Board hc1:     okay    okay    okay    okay    okay    n.a.    n.a.    okay    n.a.    n.a.    n.a.    okay    okay    okay    okay    okay    okay    done    [WARN]
Board pi:      fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    fail    n.a.    fail    done    [WARN]
mszyprow@AMDC2765:~/dev/kernel$
```

- Some boards doesn’t even boot

Simple regression – next-20220308

- Lets investigate the logs

```
13:30:13.353153 [    2.597086] Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
13:30:13.398802 [    2.614381] 13800000.serial: ttySAC0 at MMIO 0x13800000 (irq = 67, base_baud = 0) is a S3C6400/10
13:30:13.398858 [    2.623534] serial serial0: tty port ttySAC0 registered
13:30:13.398871 [    2.629157] 13810000.serial: ttySAC1 at MMIO 0x13810000 (irq = 68, base_baud = 0) is a S3C6400/10
13:30:13.398881 [    2.639971] 13820000.serial: ttySAC2 at MMIO 0x13820000 (irq = 69, base_baud = 0) is a S3C6400/10
13:30:13.398891 [    2.648297] printk: console [ttySAC2] enabled
13:30:13.398900 [    2.648297] printk: console [ttySAC2] enabled
13:30:13.398909 [    2.656272] printk: bootconsole [exynos4210] disabled
13:30:13.442907 [    2.656272] printk: bootconsole [exynos4210] disabled
13:30:13.442957 [    2.669129] 13830000.serial: ttySAC3 at MMIO 0x13830000 (irq = 70, base_baud = 0) is a S3C6400/10
13:30:13.442969 [    2.682780] exynos4-fb 11c00000.fimd: Adding to iommu group 0
13:30:13.442978 [    2.688990] OF: graph: no port node found in /soc/fimd@11c00000
13:30:13.502816 [    2.708378] [drm] Exynos DRM: using 11c00000.fimd device for DMA mapping operations
13:30:13.502876 [    2.715396] exynos-drm exynos-drm: bound 11c00000.fimd (ops fimd_component_ops)
13:30:13.502889 [    2.722280] OF: graph: no port node found in /soc/dsi@11c80000
13:30:13.502899 [    2.731295] exynos-drm exynos-drm: bound 11c80000.dsi (ops exynos_dsi_component_ops)
13:30:13.502908 [    2.739234] exynos-drm exynos-drm: [drm] Cannot find any crtc or sizes
13:30:13.502916 [    2.748932] [drm] Initialized exynos 1.1.0 20180330 for exynos-drm on minor 0
13:30:14.005874 [    3.011141] panel-samsung-s6e8aa0 11c80000.dsi.0: ID: 0x12, 0x8e, 0x9f
13:33:42.032171
13:33:42.032226 +++ test boot status fail
```

Simple regression – bisecting basics

- What do we know:
  - Base release (v5.17-rc1) boots fine
  - Linux next-20220308 fails to boot
- GIT SCM has a subcommand (‘bisect’) for finding regressions

```
$ git bisect start next-20220308 v5.17-rc1
Bisecting: 6373 revisions left to test after this (roughly 13 steps)
[1525cdb60d271d8c30b9e27f3ceb24efbacd2cbc] Merge branch 'master' of
git://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git
```

- Run the test and tell GIT SCM the result

```
$ git bisect good|bad
```

Simple regression – bisecting basics

- Testing given commit is easy if it is a matter of running a script and inspecting result
- Compiling kernel and booting the board(s) is still time consuming

- GIT SCM can run a test script and get it results via return code
  - 0 means ‘good’
  - 1 to 127 except 125 mean ‘bad’
  - 125 means ‘skip’
  - any other – abort

Automated bisection

- Do everything (configure, compile, boot) in a single test script (‘*test_boot*’)

- Automate the process:

```
$ git bisect start next-20220308 v5.17-rc1
$ git bisect run test_boot --board=trats --config=arm_exynos
...
0d03011894d23241db1a1cad5c12aede60897d5e is the first bad commit
...
bisect run success
```

- Double check it:

```
$ git checkout next-20220308
$ git revert 0d03011894d2
$ test_boot --board=trats --config=arm_exynos
...
==== tests succeeded ====
```

Issue found – what next?

```
$ git show 0d03011894d2
commit 0d03011894d23241db1a1cad5c12aede60897d5e
Author: Thomas Zimmermann <tzimmermann@suse.de>
Date:   Wed Feb 23 20:38:03 2022 +0100

    fbdev: Improve performance of cfb_imageblit()

    Improve the performance of cfb_imageblit() by manually unrolling
    the inner blitting loop and moving some invariants out. The compiler
    failed to do this automatically. This change keeps cfb_imageblit()
    in sync with sys_imagebit().

    ...

    Signed-off-by: Thomas Zimmermann <tzimmermann@suse.de>
    Reviewed-by: Javier Martinez Canillas <javierm@redhat.com>
    Acked-by: Sam Ravnborg <sam@ravnborg.org>
    Link: https://patchwork.freedesktop.org/patch/msgid/20220223193804.18636-5-tzimmermann@suse.de

    ...
```

Issue found – report it!

- **Make sure to notify everyone involved**

- Find the original mail with the faulty patch
  - Easy case – a commit with a '*Link:*' tag
    - Patchwork
    - http://lore.kernel.org
  - Otherwise, search lore.kernel.org for that patch

- **Check if the issue has been already reported**

Issue found – report it!

- Include at least the following information:
  - **What is the regression**
  - Which source tree has been tested
  - Hardware platform, device-tree file
  - Kernel architecture and configuration
  - Stack trace if such can be obtained
  - If reverting on top of next helps?

- Add anything else we have already spotted

Issue found – example report

Re: [v3,4/5] fbdev: Improve performance of cfb_imageblit()

Marek Szyprowski 08.03.2022, 23:52

Hi Thomas,

On 23.02.2022 20:38, Thomas Zimmermann wrote:

> Improve the performance of cfb_imageblit() by manually unrolling
> the inner blitting loop and moving some invariants out. The compiler
> failed to do this automatically. This change keeps cfb_imageblit()
> in sync with sys_imagebit().

...

> Signed-off-by: Thomas Zimmermann <tzimmermann@suse.de>
> Acked-by: Sam Ravnborg <sam@ravnborg.org>
> Reviewed-by: Javier Martinez Canillas <javierm@redhat.com>

This patch landed recently in linux next-20220308 as commit 0d03011894d2 ("fbdev: Improve performance of cfb_imageblit()"). Sadly it causes a freeze after DRM and emulated fbdev initialization on various Samsung Exynos ARM 32bit based boards. This happens when kernel is compiled from exynos_defconfig. Surprisingly when kernel is compiled from multi_v7_defconfig all those boards boot fine, so this is a matter of one of the debugging options enabled in the exynos_defconfig. I will try to analyze this further and share the results. Reverting $subject on top of next-20220308 fixes the boot issue.

Another regression

- Linux next-20220518
- Lets run the tests…

Another regression

- Linux next-20220518
- Lets run the tests

File Edit View Search Terminal Help

arm64:

Test:	boot	bash	defer	rtc0	rtc1	drm	fbtest	ping	wifi	bt	alsa	cpu_hp	sleep	sleep2	ping2	reboot	off	status	[summary]
Board juno:	okay	okay	fail	okay	n.a.	fail	fail	okay	n.a.	n.a.	n.a.	okay	okay	okay	okay	okay	okay	done	[WARN]
Board rpi3:	okay	okay	okay	n.a.	n.a.	okay	okay	okay	okay	fail	fail	n.a.	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board rpi4:	okay	okay	okay	n.a.	n.a.	okay	okay	okay	okay	okay	fail	n.a.	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board tm2e:	okay	okay	okay	okay	n.a.	okay	okay	okay	okay	okay	okay	okay	okay	okay	okay	okay	okay	done	[WARN]
Board c4:	okay	okay	okay	n.a.	n.a.	okay	okay	okay	n.a.	n.a.	fail	okay	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board n2:	okay	okay	okay	n.a.	n.a.	okay	okay	okay	n.a.	n.a.	fail	okay	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board vim3:	okay	okay	okay	n.a.	n.a.	okay	fail	okay	okay	okay	fail	okay	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board vim3l:	okay	okay	okay	n.a.	n.a.	okay	okay	okay	okay	okay	fail	okay	n.a.	n.a.	okay	okay	okay	done	[WARN]
Board db410c:	okay	okay	okay	okay	n.a.	okay	fail	okay	fail	fail	okay	okay	fail	fail	fail	n.a.	n.a.	done	[WARN]
Board virt:	okay	okay	okay	okay	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	okay	okay	okay	n.a.	okay	okay	done	[WARN]

- All boards boot, most tests succeeded
  - there are some warnings

Another regression – next-20220518

- Lets investigate the logs

```
16:11:25.437034 [    1.712884] printk: console [ttyS0] enabled
16:11:25.437044 [    1.712894] printk: bootconsole [uart8250] disabled
16:11:25.437055 [    1.727298] printk: bootconsole [uart8250] printing thread stopped
16:11:25.517544 [    1.778475] ------------[ cut here ]------------
16:11:25.517597 [    1.778486] WARNING: CPU: 0 PID: 1 at block/blk-cgroup.c:301 blkg_create+0x3a0/0x4f0
16:11:25.517607 [    1.791111] Modules linked in:
16:11:25.517614 [    1.794230] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.18.0-rc7-next-20220518+ #5064
16:11:25.517622 [    1.802192] Hardware name: Raspberry Pi 4 Model B (DT)
16:11:25.517629 [    1.807414] pstate: 600000c5 (nZCv daIF -PAN -UAO -TCO -DIT -SSBS BTYPE=--)
16:11:25.517636 [    1.814492] pc : blkg_create+0x3a0/0x4f0
16:11:25.517643 [    1.818490] lr : blkg_create+0x24/0x4f0
16:11:25.517650 [    1.822398] sp : ffff80000b6fbbf0
...
16:11:25.605715 [    1.898529] Call trace:
16:11:25.605742 [    1.901018]  blkg_create+0x3a0/0x4f0
16:11:25.605768 [    1.904663]  blkcg_init_queue+0x74/0x204
16:11:25.605793 [    1.908662]  __alloc_disk_node+0xf8/0x1f0
16:11:25.649576 [    1.912744]  __blk_alloc_disk+0x38/0x140
16:11:25.649636 [    1.916737]  brd_alloc.part.0+0xf8/0x220
16:11:25.649654 [    1.920729]  brd_init+0xe8/0x164
16:11:25.649668 [    1.924023]  do_one_initcall+0x74/0x400
16:11:25.649682 [    1.927928]  kernel_init_freeable+0x2f4/0x37c
16:11:25.649696 [    1.932362]  kernel_init+0x28/0x130
16:11:25.649710 [    1.935916]  ret_from_fork+0x10/0x20
```

Another automated bisection

- The warning includes a function name and an offset – '*blkg_create+0x398/0x4e0*'
  - Function name is quite unique and doesn't normally appear in the logs
  - we can use it for checking if the test succeeded or failed

- Again, automate the process:

```
$ git bisect start next-20220518 v5.18-rc1
$ git bisect run test_boot --board=rpi4 --config=arm64 --bad="blkg_create"
...
77c570a1ea85ba4ab135c61a028420a6e9fe77f3 is the first bad commit
...
bisect run success
```

- Double check it:

```
$ git checkout next-20220518
$ git revert 77c570a1ea85
$ test_boot --board=rpi4 --config=arm64 --bad="blkg_create"
...
==== tests succeeded ====
```

Another issue found – my report

Re: [PATCH] blk-cgroup: Remove unnecessary rcu_read_lock/unlock()
Marek Szyprowski 18.05.2022, 21:28

On 16.05.2022 19:39, bh1scw@gmail.com wrote:
> From: Fanjun Kong <bh1scw@gmail.com>
>
> spin_lock_irq/spin_unlock_irq contains preempt_disable/enable().
> Which can serve as RCU read-side critical region, so remove
> rcu_read_lock/unlock().
>
> Signed-off-by: Fanjun Kong <bh1scw@gmail.com>

This patch landed in today's linux next-20220518 as commit 77c570a1ea85 ("blk-cgroup: Remove unnecessary rcu_read_lock/unlock()").

Unfortunately it triggers the following warning on ARM64 based Raspberry Pi 4B board:

```
------------[ cut here ]------------
WARNING: CPU: 0 PID: 1 at block/blk-cgroup.c:301
blkg_create+0x398/0x4e0
Modules linked in:
CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.18.0-rc3+ #5080
Hardware name: Raspberry Pi 4 Model B (DT)
pstate: 600000c5 (nZCv daIF -PAN -UAO -TCO -DIT -SSBS BTYPE=--)
pc : blkg_create+0x398/0x4e0
...
Call trace:
 blkg_create+0x398/0x4e0
 blkcg_init_queue+0x74/0x204
 __alloc_disk_node+0xf8/0x1f0
 __blk_alloc_disk+0x38/0x140
 brd_alloc.part.0+0xf8/0x220
 brd_init+0xe8/0x164
 do_one_initcall+0x74/0x400
 kernel_init_freeable+0x2f4/0x37c
 kernel_init+0x28/0x130
 ret_from_fork+0x10/0x20
...
---[ end trace 0000000000000000 ]---
```

If this is a false positive, then the check in the code needs to be adjusted.

How to handle more complex issues?

- Reverting commit on top of next fails
  - Try ‘*git mergetool*’ to resolve conflict(s)
  - Find all commits that modify affected files
  - Revert them too
  - Usually affects the whole patch-series
- Example:
  - Linux next-20220331
  - Commit 57c47b42f454 ("block: turn bio_kmalloc into a simple kmalloc wrapper")

Reverting commit on top of next fails

```
$ git checkout next-20220331

$ git revert 57c47b42f454
error: could not revert 57c47b42f454... block: turn bio_kmalloc into a simple kmalloc wrapper
...

$ git mergetool
Merging:
drivers/block/pktcdvd.c
include/linux/bio.h

Normal merge conflict for 'drivers/block/pktcdvd.c':
  {local}: modified file
  {remote}: modified file
merge of drivers/block/pktcdvd.c failed

$ git reset --hard next-20220331
```

Reverting commit on top of next fails

```
$ git log --no-merges --oneline 57c47b42f454..HEAD drivers/block/pktcdvd.c
1292fb59f283 pktcdvd: stop using bio_reset
47c426d52417 pktcdvd: remove a pointless debug check in pkt_submit_bio
dbb4c84d87af scsi: core: Move the result field from struct scsi_request to struct scsi_cmnd
ce70fd9a551a scsi: core: Remove the cmd field from struct scsi_request

$ git revert 1292fb59f283
[detached HEAD 7b058243db2d] Revert "pktcdvd: stop using bio_reset"
 1 file changed, 16 insertions(+), 9 deletions(-)

$ git revert 57c47b42f454
error: could not revert 57c47b42f454... block: turn bio_kmalloc into a simple kmalloc wrapper

$ git mergetool
Merging:
include/linux/bio.h
...

$ test_boot --board=trats --config=arm_exynos
...
==== tests succeeded ====
```

Issue found and confirmed – report it!

Re: [PATCH 4/5] block: turn bio_kmalloc into a simple kmalloc wrapper

Marek Szyprowski **31.03.2022, 23:18**

Hi Christoph,

On 08.03.2022 07:15, Christoph Hellwig wrote:

> Remove the magic autofree semantics and require the callers to explicitly
> call bio_init to initialize the bio.
>
> This allows bio_free to catch accidental bio_put calls on bio_init()ed
> bios as well.
>
> Signed-off-by: Christoph Hellwig <hch@lst.de>

This patch, which landed in today's next-20220331 as commit 57c47b42f454 ("block: turn bio_kmalloc into a simple kmalloc wrapper"), breaks badly all my test systems, which use squashfs initrd:

```
RAMDISK: squashfs filesystem found at block 0
RAMDISK: Loading 2489KiB [1 disk] into ram disk... done.
using deprecated initrd support, will be removed in 2021.
------------[ cut here ]------------
WARNING: CPU: 4 PID: 1 at block/bio.c:229 bio_free+0x6c/0x70
Modules linked in:
CPU: 4 PID: 1 Comm: swapper/0 Not tainted 5.17.0-next-20220331 #4767
Hardware name: Samsung Exynos (Flattened Device Tree)
 unwind_backtrace from show_stack+0x10/0x14
 show_stack from dump_stack_lvl+0x58/0x70
 dump_stack_lvl from __warn+0xc8/0x218
 __warn from warn_slowpath_fmt+0x5c/0xb4
 warn_slowpath_fmt from bio_free+0x6c/0x70
 bio_free from squashfs_read_data+0x118/0x748
 squashfs_read_data from squashfs_read_table+0xdc/0x144
 squashfs_read_table from squashfs_fill_super+0x100/0x9ec
 squashfs_fill_super from get_tree_bdev+0x154/0x248
 get_tree_bdev from vfs_get_tree+0x24/0xe4
 vfs_get_tree from path_mount+0x3d0/0xb14
 path_mount from init_mount+0x54/0x80
 init_mount from do_mount_root+0x78/0x104
 do_mount_root from mount_block_root+0xf0/0x1fc
 mount_block_root from initrd_load+0xec/0x294
 initrd_load from prepare_namespace+0xdc/0x18c
 prepare_namespace from kernel_init+0x18/0x12c
 kernel_init from ret_from_fork+0x14/0x2c
...
Kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b
```

Reverting it on top of linux next-20220331 (together with commit 1292fb59f283 ("pktcdvd: stop using bio_reset")) fixes (or hides?) the issue.

More complex issues

- More than one new issue in a single release
  - Bisect usually finds one of them
  - Second can be found by running ‘git bisect’ again, ensuring that the first issue is always reverted
- Example:
  - Linux next-20220413,
  - Commit 33de0aa4bae9 (“genirq: Always limit the affinity to online CPUs”)
  - Commit fa6009949163 ("mm: check against orig_pte for finish_fault()")

Bisecting when there is more than one issue

- Hint for easier handling ‘a revert’, while bisecting:

```
$ git checkout next-20220413
$ git revert fa6009949163
$ git reset --mixed next-20220413
$ git stash
```

- No need to check if given change is already on the tested branch

```
$ git stash apply
```

- Applying stashed changes can be also integrated to the script

Even more complex issues

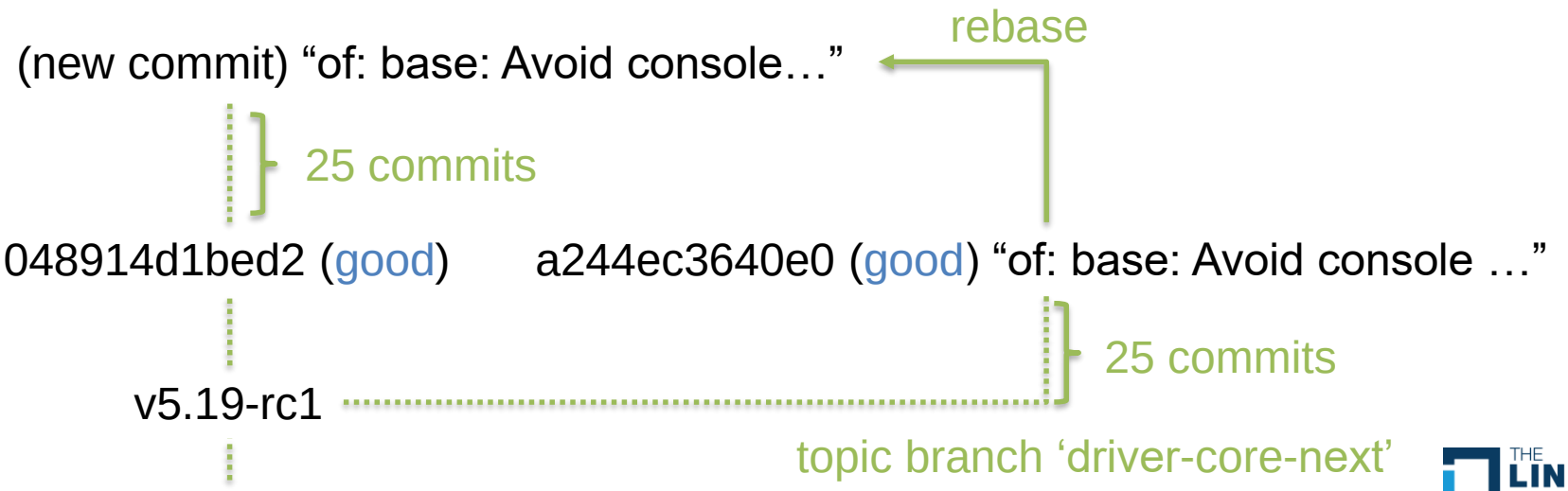
- Bisecting points to a merge commit
- There are some non-trivial dependencies between both merged branches
- Example:
  - Linux next-20220630
  - Bisecting points to commit 5732b42edfd1 ("Merge branch 'driver-core-next' …")

Bisecting points to a merge commit – next-20220630

- Merge commit 5732b42edfd1 ("Merge branch 'driver-core-next' …")
  - Main branch: 5732b42edfd1^1 = 048914d1bed2
  - Merged topic branch: 5732b42edfd1^2 = a244ec3640e0

How to cope with merge commits

- My solution - rebase the topic branch onto the main branch
- Perform the bisection again between 048914d1bed2 and new commit



How to cope with merge commits, part 2

- Second bisect points to a commit “kernfs: Change kernfs_notify_list to llist.”
  - Hash-id irrelevant, because of the rebase
  - Commit b8f35fa1188b ("kernfs: Change kernfs_notify_list to llist.") on the topic branch

- Reverting it on top of the mentioned merge commit fixed the issue
- Another check – reverting on top of next-20220630 also fixed the issue
- Report it!

Yet another possible problem

- Bisecting through broken code
  - Sometimes the code doesn’t even compile
- Ensure that the script for the automated bisecting aborts, not report it as failure
  - Better to analyze it manually than see a false result
- You may try to ignore that set of commits:

```
$ git bisect skip
```

A little summary

- Presented some of my solutions for finding the kernel regressions
- I was really surprised how many issues I've found
- A script with a simple pattern search in the logs covers most of the bisecting cases

- Results:

```
$ git log --grep="Reported-by:.*m.szyprowski" --oneline v5.19 | wc -l
117
$ git log --grep="Tested-by:.*m.szyprowski" --oneline v5.19 | wc -l
308
```

A little summary, part 2

- There are false positives, though:
  - Common console for kernel logs and user-space (mixed logs confuses *expect* tool)
  - Test hardware relies on the USB devices (UART, relays control, CDC Ethernet)

A little summary, part 3

- A lots of the issues have been found because of the debugging options enabled in exynos_defconfig:
  - CONFIG_PROVE_LOCKING
  - CONFIG_DEBUG_ATOMIC_SLEEP
  - CONFIG_PM_DEBUG
  - CONFIG_PM_ADVANCED_DEBUG
- It is worth enabling them always when doing regression tests
- It is still a bit of manual work…
- Analysing regressions is a nice hobby and mental exercise ☺

Other public Linux kernel testing solutions

- I’m not alone in testing the Linux kernel

- Kernel CI – http://kernelci.org/
- 0-DAY CI Kernel Test Service (linux test robot) – https://www.intel.com/content/www/us/en/developer/topic-technology/open/linux-kernel-performance/overview.html
- Others

- Remember – you will never be faster than any of the robots ☺

Q&A

- Questions?
- You can always mail me:
  m.szyprowski@samsung.com

EMBEDDED LINUX CONFERENCE

@

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

EUROPE

THE LINUX FOUNDATION